Epitope prediction improved by multitask support vector machines

Laurent Jacob
Centre for Computational Biology
Ecole des Mines de Paris
35, rue Saint-Honoré
77300 Fontainebleau, France
laurent.jacob@ensmp.fr

Jean-Philippe Vert
Centre for Computational Biology
Ecole des Mines de Paris
35, rue Saint-Honoré
77300 Fontainebleau, France
jean-philippe.vert@ensmp.fr

February 7, 2008

Abstract

In silico methods for the prediction of antigenic peptides binding to MHC class I molecules play an increasingly important role in the identification of T-cell epitopes. Statistical and machine learning methods, in particular, are widely used to score candidate epitopes based on their similarity with known epitopes and non epitopes. The genes coding for the MHC molecules, however, are highly polymorphic, and statistical methods have difficulties to build models for alleles with few known epitopes. In this case, recent works have demonstrated the utility of leveraging information across alleles to improve the performance of the prediction.

We design a support vector machine algorithm that is able to learn epitope models for all alleles simultaneously, by sharing information across similar alleles. The sharing of information across alleles is controlled by a user-defined measure of similarity between alleles. We show that this similarity can be defined in terms of supertypes, or more directly by comparing key residues known to play a role in the peptide-MHC binding. We illustrate the potential of this approach on various benchmark experiments where it outperforms other state-of-the-art methods.

1 Introduction

A key step in the immune response to pathogen invasion is the activation of cytotoxic T-cells, which is triggered by the recognition of a short peptide, called epitope, bound to Major Histocompatibility Complex (MHC) class I molecules and presented to the T-cells. This recognition is supposed to trigger cloning and activation of cytotoxic lymphocytes able to identify and destroy the pathogen or infected cells. MHC class I epitopes are therefore potential tools for the development of peptide vaccines, in particular for AIDS vaccines (McMichael and Hanke, 2002). They are also potential tools for diagnosis and treatment of cancer (Wang, 1999; Sette *et al.*, 2001).

Identifying MHC class I epitope in a pathogen genome is therefore crucial for vaccine design. However, not all peptides of a pathogen can bind to the MHC molecule to be presented to T-cells: it is estimated that only 1 in 100 or 200 peptides actually binds to a particular MHC (Yewdell and Bennink, 1999). In order to alleviate the cost and time required to identify epitopes experimentally, *in silico* computational methods for epitope prediction are therefore increasingly used. Structural approaches, on the one hand, try to evaluate how well a candidate epitope fit in the binding groove of a MHC molecule, by various threading or docking approaches (Rosenfeld *et al.*, 1995; Schueler-Furman *et al.*, 2000; Tong *et al.*, 2006; Bui *et al.*, 2006). Sequence-based approaches, on the other hand, estimate predictive models for epitopes by analyzing and learning from sets of known epitopes and non-epitopes. Models can be based on motifs (Rötzschke *et al.*, 1992; Rammensee *et al.*, 1995), profiles (Parker *et al.*, 1994; Rammensee *et al.*, 1999; Reche *et al.*, 2002), or machine learning methods like artificial neural networks (Honeyman *et al.*, 1998; Milik *et al.*, 1998; Brusic *et al.*, 2002; Buus *et al.*, 2003; Nielsen *et al.*, 2003; Zhang *et al.*, 2005), hidden Markov models (Mamitsuka, 1998), support vector machines (SVM) (Dönnes and Elofsson, 2002; Zhao *et al.*, 2003; Bhasin and Raghava, 2004; Salomon and Flower, 2006), boosted metric learning (Hertz and Yanover, 2006) or logistic regression (Heckerman *et al.*, 2006). Finally, some authors have recently proposed to combine structural and sequence-based approaches (Antes *et al.*, 2006; Jojic *et al.*, 2006). Although comparison is difficult, sequence-based approaches that learn a model from the analysis of known epitopes benefit from the accumulation of experimentally validated epitopes and will certainly continue to improve as more data become available.

The binding affinity of a peptide depends on the MHC molecule's 3D structure and physicochemical properties, which in turns vary between MHC alleles. This compels any prediction method to be allele-specific: indeed, the fact that a peptide can bind to an allele is neither sufficient nor necessary for it to bind to another allele. Since MHC genes are highly polymorphic, little training data if any is available for some alleles. Thus, though achieving good precisions in general, classical statistical and machine learning-based MHC-peptide binding prediction methods fail to efficiently predict bindings for these alleles.

Some alleles, however, can share binding properties. In particular, experimental work (Sidney *et al.*, 1995, 1996; Sette and Sidney, 1998, 1999) shows that different alleles have overlapping peptide repertoires. This fact, together with the posterior observation of structural similarities among the alleles sharing their repertoires allowed the definition of HLA allele supertypes, which are families of alleles exhibiting the same behavior in terms of peptide binding. This suggests that sharing information about known epitopes across different but similar alleles has the potential to improve predictive models by increasing the quantity of data used to establish the model. For example, Zhu *et al.* (2006) show that simply pooling together known epitopes for different alleles of a given supertype to train a model can improve the accuracy of the model. Hertz and Yanover (2006) pool together epitope data for all alleles simultaneously to learn a metric between peptides, which is then used to build predictive models for each allele. Finally, Heckerman *et al.* (2006) show that leveraging the information across MHC alleles and supertypes considerably improves individual allele prediction accuracy.

In this paper we show how this strategy of leveraging information across different alleles when learning allele-specific epitope prediction models can be naturally performed in the context of SVM, a state-of-the-art machine learning algorithm. This new formulation is based on the notion of *multitask kernels* (Evgeniou *et al.*, 2005), a general framework for solving several related machine learning problems simultaneously. Known epitopes for a given allele contribute to the model estimation for all other alleles, with a weight that depends on the similarity between alleles. Here the notion of similarity between alleles can be very general; we can for example follow Heckerman *et al.* (2006) and define two alleles to be similar if they belong to the same supertype, but the flexibility of our mathematical formulation also allows for more subtle notions of similarity, based for example of sequence similarity between alleles. On a benchmark experiment we demonstrate the relevance of the multitask SVM approach which outperforms state-of-the-art prediction methods.

2 Methods

In this section, we explain how information can be shared between alleles when SVM models are trained on different alleles. For the sake of clarity we first explain the approach in the case of linear classifiers, and then generalize it to more general models.

2.1 Sharing information with linear classifiers

Let us first assume that epitopes are represented by d-dimensional vectors x, and that for each allele a we want to learn a linear function $f_a(x) = w^\top x$ to discriminate between epitopes and non-epitopes, where $w \in \mathbb{R}^d$. A natural way to share information between different alleles is to assume that each vector w is the sum of a common vector w_c which is common to all alleles, and of an allele-specific vector w_a, resulting in a classifier:

$$f_a(x) = (w_c + w_a)^\top x \,. \tag{1}$$

In this equation the first term w_c accounts for general characteristics of epitopes valid for all alleles, while the second term w_a accounts for allele-specific properties of epitopes. In order to estimate such a model from data it is convenient to rewrite it as a simple linear model in a larger space as follows. Assuming that there are p alleles $\{a_1, \ldots, a_p\}$ we can indeed rewrite (1) as:

$$f_a(x) = W^\top \Phi(a, x)\,, \tag{2}$$

where W is the $d \times (p+1)$-dimensional vector $W = \left(w_c^\top, w_{a_1}^\top, \ldots, w_{a_p}^\top \right)^\top$ and $\Phi(a, x) = (x^\top, 0^\top, \ldots, 0^\top, x^\top, 0^\top, \ldots, 0^\top)^\top \in \mathbb{R}^{d \times (p+1)}$ is the vector obtained by concatenating the vector x with p blocks of zeros, except for the a-th block which is a copy of x. Indeed it is then easy to check that $W^\top \Phi(a, x) = (w_c + w_a)^\top x$, hence that (1) and (2) are equivalent. Each (peptide,allele) pair is therefore mapped to a large vector $\Phi(x, a)$ with only two non-zero parts, one common to all alleles and one at an allele-specific position.

The parameters of this model, namely the weights w_c and w_a for all alleles a, can then be learned simultaneously by any linear model, such as logistic regression or SVM, that estimates a vector W in (2) from a training set $((x_1, a_1, y_1), \ldots, (x_n, a_n, y_n))$ of (peptide,allele) pairs labeled as $y_i = +1$ if peptide x_i is an epitope of allele a_i, $y_i = -1$ otherwise. This approach was followed by Heckerman *et al.* (2006) who included another level of granularity to describe how information is shared across alleles, by considering allele-specific, supertype-specific and common weight vectors.

In summary, it is possible to embed the allele information in the description of the data point to estimate linear models in the new peptide × allele space to share information across alleles. It is furthermore possible to adjust how information is shared by choosing adequate functions $\Phi(x, a)$ to represent (peptide,allele) pairs. In other words, it is possible to consider the problem of leveraging across the alleles as a simple choice of representation, or feature design for the (peptide,allele) pairs that are to be used to learn the classifier. This approach, however, is limited by at least two constraints:

- It can be uneasy to figure out how to represent the allele information in the mapping $\Phi(x, a)$. In Heckerman *et al.*, 2006, this is done *via* Boolean conjunctions and leads to a convenient form for the prediction functions, like (1) with a third term accounting for the supertype. Including more prior knowledge regarding when two alleles should share more information, *e.g.*, based on structural similarity between alleles, is however not an easy task.

- Practically, injecting new features in the vector $\Phi(x, a)$ increases the dimension of the space, making statistical estimation, storage, manipulation and optimization tasks much harder.

In the next subsection we show how both limitations can be overcome by reformulating this approach in the framework of kernel methods.

2.2 The kernel point of view

SVM, and more generally kernel methods, only access data through the computation of inner products between pairs of data points, called a *kernel* function (Vapnik, 1998; Schölkopf and Smola, 2002; Schölkopf *et al.*, 2004). As a result, estimating the weights W in (2) with a SVM does not require to explicitly compute or store the vectors $\Phi(x,a)$ for training and test pairs of alleles and peptides. Instead, it only requires to be able to compute the kernel between any two pairs (x,a) and (x',a') given, in our linear example, by:

$$\begin{aligned} K\left((x,a),(x',a')\right) =& \Phi(x,a)^\top \Phi(x',a') \\ =& \begin{cases} 2x^T x' & \text{if } a = a'\,, \\ x^T x' & \text{if } a \neq a'\,. \end{cases} \end{aligned}$$

Let us now introduce the following two kernels, respectively between peptides only and between alleles only:

$$\begin{aligned} K_{pep}(x,x') &\triangleq x^\top x' \\ K_{all}(a,a') &\triangleq \begin{cases} 2 & \text{if } a = a'\,, \\ 1 & \text{if } a \neq a'\,. \end{cases} \end{aligned}$$

It is easy to see that both kernels are valid positive definite kernels for peptides and alleles, respectively. With these notations we see that the kernel for pairs (x,a) can be expressed as the *product* of the kernel for alleles and the kernel for peptides:

$$K\left((x,a),(x',a')\right) = K_{all}(a,a')K_{pep}(x,x')\,, \tag{3}$$

which is also the kernel associated to the tensor product space of the Hilbert spaces associated to K_{pep} and K_{all} (Aronszajn, 1950). Such kernels are used in particular in the field of *multitask learning* Evgeniou *et al.* (2005), where several related machine learning tasks must be solved simultaneously. The allele kernel K_{all} quantifies how information is shared between alleles. For example, in the simple model (1) the kernel is simply equal to 2 if an allele is compared to itself, 1 otherwise, meaning that information is uniformly shared across different alleles. Alternatively, adding supertype-specific features like Heckerman *et al.* (2006) would result in a kernel equal to 3 between an allele and itself, 2 between two different alleles that belong to a common supertype, and 1 otherwise, resulting in increased sharing of information within supertypes.

Interestingly this formulation lends itself particularly well to further generalization. Indeed, for any positive definite kernels K_{all} and K_{pep} for alleles and peptides, respectively, their product (3) is a valid positive definite kernel over the product space of pairs (peptide,allele) (Aronszajn, 1950). This suggests a new strategy to design predictive models for epitopes across alleles, by designing specific kernels for alleles and peptides, respectively, and combining them to learn all allele-specific models simultaneously with the tensor product kernel (3). Benefits of this strategy over the explicit design and computation of feature vectors $\Phi(x, a)$ are two-folds. First, it splits the problem of feature vector design into two subproblems (designing two kernels), each of which can benefit from previous work on kernel design (*e.g.*, Schölkopf *et al.*, 2004). For example, the fact that nonlinear kernels such as Gaussian or polynomial kernels for peptides give good results for SVM trained on individual alleles suggest that they are natural candidates for the peptide part of the product kernel. Second, working with kernels alleviates the practical issues due to the potentially large size of the feature vector representation $\Phi(x, a)$ in terms of memory for storage or speed of convergence of algorithms. We now describe in more details the kernels K_{pep} and K_{all} that can be used for peptides and alleles, respectively, to create the product kernel used in the application.

2.3 Peptide kernels

We consider in this paper mainly peptides made of 9 amino acids, although extensions to variable-length peptides poses no difficulty in principle (Salomon and Flower, 2006). The classical way to represent these 9-mers as fixed length vectors is to encode the letter at each position by a 20-dimensional binary vector indicating which amino acid is present, resulting in a 180-dimensional vector representations. In terms of kernel, the inner product between two peptides in this representation is simply the number of letters they have in common at the same positions, which we take as our baseline kernel:

$$K_{linseq}(x, x') = \sum_{i=1}^{l} \delta(x[i]x'[i]),$$

where l is the length of the peptides (9 in our case), $x[i]$ is the i-th residue in x and $\delta(x[i]x'[i])$ is 1 if $x[i] = x'[i]$, 0 otherwise.

Alternatively, several authors have noted that nonlinear variants of the linear kernel can improve the performance of SVM for epitope prediction (Dönnes and Elofsson, 2002; Zhao *et al.*, 2003; Bhasin and Raghava, 2004). In particular, using a polynomial kernel of degree p over the baseline kernel is equivalent, in terms of feature space, to encoding p-order interactions between amino acids at different positions. In order to assess the relevance of such non-linear extensions we tested a polynomial kernel of degree 5, *i.e.*,

$$K_{seq5}(x, x') = (K_{linseq}(x, x') + 1)^5.$$

In order to limit the risk of overfitting to the benchmark data we restrict ourselves to the evaluation of the baseline linear kernel and its nonlinear polynomial extension. Designing a specific peptide kernel for epitope prediction, *e.g.*, by weighting differently the positions known to be critical in the MHC-peptide complex, is however an interesting research topic that could bring further improvements in the future.

2.4 Allele kernels

Although the question of kernel design for peptides has been raised in previous studies involving SVM for epitope prediction (Dönnes and Elofsson, 2002; Zhao *et al.*, 2003; Bhasin and Raghava, 2004; Salomon and Flower, 2006), the question of kernel design for alleles is new to our knowledge. We tested several choices that correspond to previously published approaches:

- The *Dirac* kernel is:

  $$K_{Dirac}(a, a') = \begin{cases} 1 & \text{if } a = a' , \\ 0 & \text{otherwise.} \end{cases}$$

  With the Dirac kernel, no information is shared across alleles and the SVM learns one model for each allele independently from the others. Therefore this corresponds to the classical setting of learning epitope prediction models per allele with SVM.

- The *uniform* kernel is:

  $$K_{uniform}(a, a') = 1 \text{ for all } a, a' .$$

  With this kernel all alleles are considered the same, and a unique model is created by pooling together the data available for all alleles.

- The *multitask* kernel is:

  $$K_{multitask}(a, a') = K_{dirac}(a, a') + K_{uniform}(a, a') .$$

  As explained in the previous section and in Evgeniou *et al.* (2005) this is the simplest way to train different but related models. The SVM learns one model for each allele, using known epitopes and non-epitopes for the allele, but using also known epitopes and non-epitope for all other alleles with a smaller contribution. The training peptides are shared uniformly across different alleles.

- The *supertype* kernel is

  $$K_{supertype}(a, a') = K_{multitask} + \delta_s(a, a') ,$$

  where $\delta_s(a, a')$ is 1 if a and a' are in the same supertype, 0 otherwise. As explained in the previous section this scheme trains a specific models for each allele using training peptides from different alleles, but here the training peptides are more shared across alleles withing a supertype than across alleles in different supertypes. This is used by Heckerman *et al.* (2006), without the kernel formulation, to train a logistic regression model.

Heckerman *et al.* (2006) show that the supertype kernel generally improves the performance of logistic regression models compared to the uniform or Dirac kernel. Intuitively it seems to be an interesting way to include prior knowledge about alleles. However, one should be careful since the definition of supertypes is based on the comparison of epitopes of different alleles, which suggests that the supertype information might be based on some information used to assess the performance of the method in the benchmark experiment. In order to overcome this issue, and illustrate the possibilities offered by our formulation, we also tested a kernel between alleles which tries to quantify the similarity of alleles without using known epitope information. For that purpose we reasoned that alleles with similar residues at the positions involved in the peptide binding were more likely to have similar epitopes, and decided to make a kernel between alleles based on this information. For each locus we gathered from Doytchinova *et al.* (2004) the list of positions involved in the binding site of the peptide (Table 1). Taking the union of these sets of positions we then represented each allele by the list of residues at these positions, and used a polynomial kernel of degree 7 to compare two lists of residues associated to two alleles, *i.e*,

$$K_{bsite7}(a, a') = \left(\sum_{i \in \text{bsite}} \delta(a[i]a'[i]) + 1 \right)^7 ,$$

where bsite is the set of residues implied in the binding site for one of the three allele groups HLA-A, B, C, $a[i]$ is the i-th residue in a and $\delta(a[i]a'[i])$ is 1 if $a[i] = a'[i]$, 0 otherwise.

2.5 SVM

We learn epitope models with SVM, a state-of-the-art algorithm for pattern recognition (Vapnik, 1998; Schölkopf and Smola, 2002; Schölkopf *et al.*, 2004). We used the `libsvm` SVM implementation, with a custom kernel to account for the various kernels we tested, in the PyML environment (`http://pyml.sourceforge.org`). Besides the kernel, SVM depends on one parameter usually called C. For each experiment, we selected the best C among the values $2^i, i \in \{-15, -14, \ldots, 9, 10\}$ by selecting the value leading to the largest area under the ROC curve estimated by cross-validation on the training set only. The performance of each method was then tested on each experiment by evaluating the AUC over the test data.

3 Data

In order to evaluate both the performance of our method and the impact of using various kernels for the peptides or the alleles, we test our method on three different benchmark datasets that have been compiled recently to compare the performance of epitope prediction algorithms.

We first use two datasets compiled by Heckerman *et al.* (2006), where it is already shown that leveraging improves prediction accuracy with respect to the best published results.The first dataset, called SYFPEITHY+LANL, combines experimentally confirmed positive epitopes from the SYFPEITHY database (see Rammensee *et al.*, 1999, available at `http://www.syfpeithy.de`) and from the Los Alamos HIV database (`http://www.hiv.lanl.gov`) and negative example randomly drawn from the HLA and amino acid distribution in the positive examples, for a total of 3152 data points. For more details, see Heckerman *et al.* (2006) where this dataset is used to compare the leveraged logistic regression with *DistBoost*. Since this dataset is quite small and was already used as a benchmark, we use it as a first performance evaluation, and to compare our kernels.

The second dataset of Heckerman *et al.* (2006) contains $160,085$ peptides including those from SYSFPEITHY+LANL and others from the MHCBN data repository (see Bhasin *et al.*, 2003, available at `http://www.imtech.res.in/raghava/mhcbn/index.html`). This corresponds to $1,585$ experimentally validated epitopes, and $158,500$ randomly generated non-binders (100 for each positive). We only kept 50 negative for each positive in the interest of time and assuming this would not deteriorate too much the performance of our algorithm. In the worst case, it is only a handicap for our methods.

Finally, we assess the performance of our method on the MHC-peptide binding benchmark recently proposed by Peters *et al.* (2006) who gathered quantitative peptide-binding affinity measurements for various species, MHC class I alleles and peptide lengths, which makes it an excellent tool to compare MHC-peptide binding learning methods. Since our method was first designed for binary classification of HLA epitopes, we focused on the 9-mer peptides for the 35 human alleles and thresholded at IC50 $=500$. Nevertheless, the application of our method to other species or peptide lengths would be straightforward, and generalization to quantitative prediction should not be too problematic either. The benchmark contained 29336 9-mer.

The first dataset is 5-folded, the second 10-folded, so that the test be only performed on HIV (LANL) data. The third dataset is 5-folded. We used the same folds as Heckerman *et al.* (2006), available at `ftp://ftp.research.microsoft.com/users/heckerma/recom` for the first two datasets and the same folds as Peters *et al.* (2006) available at `http://mhcbindingpredi` for the third one.

Molecule-based allele kernels require the amino-acid sequences corresponding to each allele. These sequences are available in various databases, including`http://www.anthonynolan.org` and Robinson *et al.* (2000). We used the peptide-sequence alignment for HLA-A, HLA-B and HLA-C loci. Each sequence was restricted to residues at positions involved in the binding site of one of the three loci, see table 1. Preliminary experiments showed that using this restriction instead of the whole sequences didn't change the performance significantly, but it speeds up the calculation of the kernel. We were not able to find the sequence of a few molecules of the two datasets of Heckerman *et al.* (2006), so in the experiments implying these datasets and a molecule-based allele kernel, we used $K_{bsite7}(a,a') + K_{multitask}(a,a')$ instead of simply using $K_{bsite7}(a,a')$, with a sentinel value of $K_{bsite7}(a,a') = 0$ in these cases. This is the sum of two kernels, so still a

Locus	Positions
HLA-A	5, 7, 9, 24, 25, 34, 45, 59, 63, 66, 67, 70, 74, 77, 80, 81, 84, 97, 99, 113, 114, 116, 123, 133, 143, 146, 147, 152, 155, 156, 159, 160, 163, 167, 171
HLA-B	5, 7, 8, 9, 24, 45, 59, 62, 63, 65, 66, 67, 70, 73, 74, 76, 77, 80, 81, 84, 95, 97, 99, 114, 116, 123, 143, 146, 147, 152, 155, 156, 159, 160, 163, 167, 171
HLA-C	5, 7, 9, 22, 59, 62, 64, 66, 67, 69, 70, 73, 74, 77, 80, 81, 84, 95, 97, 99, 116, 123, 124, 143, 146, 147, 156, 159, 163, 164, 167, 171

Table 1: Residue positions involved in the binding site for the three loci, according to Doytchinova *et al.* (2004)

positive definite kernel and actually exactly the same thing as $K_{supertype}$ with K_{bsite7} instead of δ_s.

4 Results

We first use K_{linseq} and K_{seq5} for the peptides and $K_{uniform}$ (one SVM for all the alleles), K_{Dirac} (one SVM for each allele), $K_{multitask}$, $K_{supertype}$ and K_{bsite7} for the alleles on the small SYFPEITHI+LANL dataset. Using combinations of molecule-based and non-molecule-based kernels for K_{all} didn't improve the prediction, generally the result was as good as or slightly worse than the result obtained with the best of the two combined kernels. Results are displayed on Table 2, and ROC curves for $K_{linseq} \times K_{Dirac}$, $K_{linseq} \times K_{supertype}$, $K_{seq5} \times K_{supertype}$ and $K_{seq5} \times K_{bsite7}$ on figure 1.

Table 2 demonstrates the benefits of carefully sharing information across alleles. The *Dirac* allele kernel being the baseline kernel corresponding to independent training of SVM on different alleles, we observe an improvement of at least 2% when information is shared across alleles during training (with the *multitask,supertype* or *bsite7* strategies). It should be noted, however, that the *uniform* strategies which amount to training a single model for all alleles perform considerably worse than the *Dirac* strategies, justifying the fact that it is still better to build individual models than a single model for all alleles. Among the strategies to share information across alleles, the *supertype* allele kernel seems to work slightly better than the two other ones. However, one should keep in mind that there is a possible bias in the performance of the *supertype* kernel, because some peptides in the test sets might have contributed to the definition of the allele supertypes. Among the *multitask* kernel, which considers all different alleles as equally similar, and the *bsite7* kernel, which shares more information between alleles that have similar residues at key positions, we observe a slight benefit for the *bsite7* kernel, which justifies the idea that including biological knowledge in our framework is simple and powerful. Finally, we observe that for all allele kernels, the

$K_{all} \backslash K_{pep}$	linseq	seq5
uniform	0.826 ± 0.010	0.883 ± 0.011
Dirac	0.891 ± 0.014	0.893 ± 0.024
multitask	0.910 ± 0.008	0.936 ± 0.008
supertype	0.923 ± 0.011	0.943 ± 0.015
bsite7	0.919 ± 0.011	0.943 ± 0.009

Table 2: AUC results for an SVM trained on the SYFPEITHI+LANL with various kernel and estimated error on the 5 folds.

nonlinear *seq5* peptide kernel outperforms the baseline *linseq* kernel, confirming that linear models based on position-specific score matrices might be a too restrictive set of models to predict accurately epitopes.

In terms of absolute value, all three allele kernels that share information across alleles combined with the nonlinear *seq5* peptide kernel (AUC $= 0.943 \pm 0.015$) strongly outperform the leveraged logistic regression of Heckerman *et al.* (2006) (AUC $= 0.906 \pm 0.016$) and the boosted distance metric learning algorithm of Hertz and Yanover (2006) (AUC $= 0.819 \pm 0.055$). This corresponds to a decrease of roughly 40% of the area above the ROC curve compared to the best method. As the boosted distance metric learning approach was shown to be superior to a variety of state-of-the-art other methods by Hertz and Yanover (2006), this suggest that our approach can compete if not overcome the best methods in terms of accuracy.

As we can clearly see in Table 2, two factors are involved in the improvement over the leveraged logistic regression of Heckerman *et al.* (2006):

- The use of an SVM instead of a logistic regression, since this is the only difference between the leveraged logistic regression and our SVM with a $K_{linseq} \times K_{supertype}$ kernel. This, however, may not be intrinsic to the algorithms, but caused by optimization issues for the logistic regression in high dimension.

- The use of a non-linear kernel for the peptide, as we observe a clear improvement in the case of SVM (this improvement might therefore also appear if the logistic regression was replaced by a kernel logistic regression model with the adequate kernel).

Figure 1 illustrates the various improvement underlined by this experiment: first from the individual SVM ($K_{linseq} \times K_{Dirac}$), to the $K_{linseq} \times K_{supertype}$ SVM which is the SVM equivalent of leveraged logistic regression, and finally to $K_{seq5} \times K_{supertype}$ and $K_{seq5} \times K_{bsite7}$ SVM that both give better performances than $K_{linseq} \times K_{supertype}$ SVM because they use a nonlinear kernel to compare the peptides. It is also worth noting that the *supertype* and the *bsite7* strategies give very similar results, which makes them two good strategies to leverage efficiently across the alleles with different information.

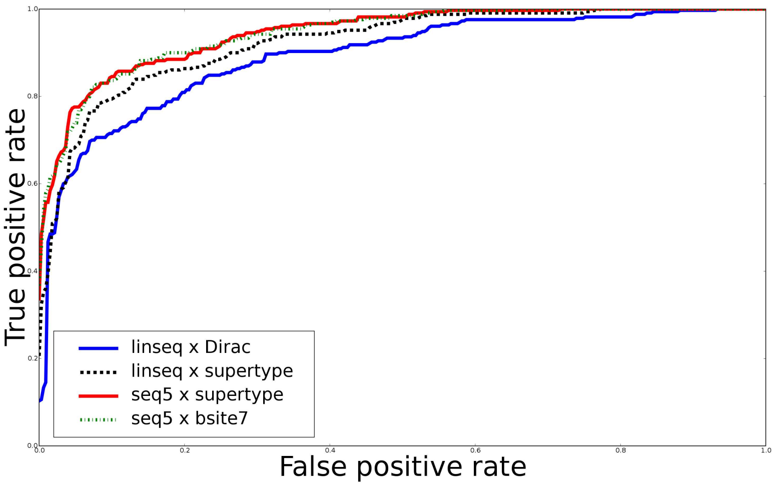

Figure 1: ROC curves on the pooled five folds of the SYFPEITHI+LANL benchmark.

These results are confirmed by the MHCBN+SYFPEITHI+LANL benchmark, for which the results are displayed in Table 3. Again, the use of SVM with our product kernels clearly improves the performance with respect to Heckerman *et al.* (2006) (from 0.906 to 0.938). Moreover, we again observe that learning a leveraged predictor using the data from all the alleles improves the global performance very strongly, hence the important step between Dirac (0.867) and all the multitask-based methods, including the simplest multitask kernel (0.934). It is worth reminding here that the multitask kernel is nothing but the sum of the Dirac and uniform kernels, *i.e.*, that it contains no additional biological information: the improvement is caused by the mere fact of using roughly (with a pondering of 0.5) the points of other alleles to learn the predictor of one allele. Figure 2 show the ROC curves for SVM with $K_{seq5} \times K_{Dirac}$, $K_{seq5} \times K_{supertype}$ and $K_{seq5} \times K_{bsite7}$ kernels on this benchmark. Again, we clearly see the strong improvement between leveraged and non-leveraged strategies. The difference between the $K_{seq5} \times K_{Dirac}$ and the two others is only caused by leveraging, since in the three case the same nonlinear strategy was used for the peptide part. On the other hand, the figure illustrates once again that our two high-level (*i.e.*, more sophisticated than *multitask*) strategies for leveraging across alleles give almost the same result.

Finally, Table 4 presents the performance on the IEDB benchmark proposed in Peters *et al.* (2006). The indicated performance corresponds, for each method, to the average on the AUC for each of the 35 alleles. This gives an indication of the global performances of each methods. The ANN field is the tool proposed in Peters *et al.* (2006) giving the best results on the 9-mer dataset, an artificial neural network proposed in Nielsen *et al.* (2003), while the ADT field refers to the adaptive double threading approach recently proposed in Jojic *et al.* (2006) and tested on the same benchmark. These tools were

Method	AUC
Leveraged LR	0.906
$K_{linseq} \times K_{stype}$	0.916 ± 0.008
$K_{seq5} \times K_{dirac}$	0.867 ± 0.010
$K_{seq5} \times K_{multitask}$	0.934 ± 0.006
$K_{seq5} \times K_{stype}$	0.939 ± 0.006
$K_{seq5} \times K_{bsite7}$	0.938 ± 0.006

Table 3: AUC results for an SVM trained on the MHCBN+SYFPEITHI+LANL benchmark with various kernel and estimated error on the 10 folds.

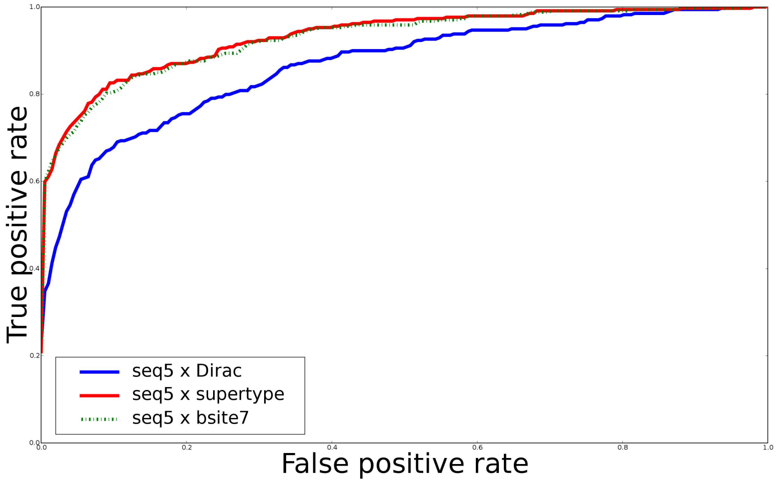

Figure 2: ROC curves on the pooled ten folds of the MHCBN+SYFPEITHI+LANL benchmark.

compared to and significantly outperformed other tools in the comprehensive study of Peters *et al.* (2006), specifically Peters and Sette (2005) and Bui *et al.* (2005), that are both scoring-matrix-based. Our approach gives equivalent results in terms of global performances as Nielsen *et al.* (2003), and therefore outperforms the other internal methods.

Table 5 presents the performances on the 10 alleles with less than 200 training points, together with the performances of the best internal tool, Nielsen *et al.* (2003) ANN, and the adaptive double threading model that gave good prediction performances on the alleles with few training data. Except for one case, our SVM outperforms both models. This means of course that our approach does not perform as well as Nielsen *et al.* (2003) on the alleles with a large training set, but nothing prevents an immunologist from using one tool for some alleles and another tool for other alleles. As we said in introduction, our original concern was to improve binding prediction for

Method	AUC
SVM with $K_{seq5} \times K_{Dirac}$	0.804
SVM with $K_{seq5} \times K_{supertype}$	0.877
SVM with $K_{seq5} \times K_{bsite7}$	0.892
ADT	0.874
ANN	0.897

Table 4: AUC results for an SVM trained on the IEDB benchmark with various methods.

Allele	Peptide number	$K_{seq5} \times K_{bsite7}$	ADT	ANN
A_2301	104	0.887 ± 0.021	0.804	0.852
A_2402	197	0.826 ± 0.025	0.785	0.825
A_2902	160	0.948 ± 0.015	0.887	0.935
A_3002	92	0.826 ± 0.048	0.763	0.744
B_1801	118	0.866 ± 0.020	0.869	0.838
B_4002	118	0.796 ± 0.025	0.819	0.754
B_4402	119	0.782 ± 0.084	0.678	0.778
B_4403	119	0.796 ± 0.042	0.624	0.763
B_4501	114	0.889 ± 0.029	0.801	0.862
B_5701	59	0.938 ± 0.046	0.832	0.926

Table 5: Detail of the IEDB benchmark for the 10 alleles with less than 200 training points (9-mer data).

alleles with few training points, and for which it is hard to generalize. This was the main point of using a multitask learning approach. The results on this last benchmark suggest that the leveraging approaches succeed in improving prediction performances when few training points are available.

5 Discussion and concluding remarks

In this paper, we introduced a general framework to share efficiently the binding information available for various alleles by simply defining a kernel for the peptides, and another one for the alleles. The result is a simple model for MHC-peptide binding prediction that uses information from the whole dataset to make specific prediction for any of the alleles. Our approach is simple, general and both easy to adapt to a specific problem by using more adequate kernels, and to implement, by running any SVM implementation with these kernels. Everything is performed in low dimension and with no need for feature selection.

We presented performances on three benchmarks. On the first two benchmark, our approach performed considerably better than the state-of-the-art, which illustrates the good general behavior in terms of prediction accuracy. Besides, these experiments clearly confirmed the interest of leveraging the information across the alleles. On the last benchmark, the results were globally comparable to the best state-of-the-art tested in Peters *et al.* (2006), with a strong improvement on the alleles for which few training points were available, probably, as it was already observed, because of the fact that our model uses all the points from all the alleles for each allele-specific prediction.

Another contribution is the use of allele sequences, which allows us to improve the prediction accuracy and to do as well as what was done with the supertype information. Supertype is a crucial information and a key concept in the development of epitope-based vaccines, for example to find epitopes that bind several alleles instead of just one. However, one should be careful when using it to learn an automatic epitope predictor because even if the idea behind a supertype definition is to represent a general ligand trend, the intuition is always guided by the fact that some alleles have overlapping repertoires of known binders, and it is not easy to figure out to which extent the known epitopes used to assess the predictor performances were used to design the supertypes.

Because of these overfitting issues and the fact that supertypes are difficult to define, the good performances of molecule-based allele kernel with respect to the supertype-based allele kernels are good news. This potentially allows us to leverage efficiently across alleles even when the supertype is unknown, which is often the case, and we don't take the risk to use overfitted information when learning on large epitope databases.

Although the kernels we used already gave good performances, there is still room for improvement. A first way to improve the performances would be to use more adequate kernels to compare the peptides and, probably more important, to compare the alleles. In other words answering the question, what does it mean in the context of MHC-peptide binding prediction for two alleles to be similar? Possible answers should probably involve better kernels for the allele sequences, and structural information which could be crucial to predict binding and, as we said in introduction, is already used in some models. Another interesting possibility is, as it was suggested in Hertz and Yanover (2007), the use of true non-binders, that could make the predictor more accurate than randomly generated peptides since these experimentally assessed peptides are in general close to the known binders. Finally, it could be useful to incorporate the quantitative IC50 information when available, instead of simply thresholding as we did for the last benchmark.

This leads us to the possible generalizations we hope to work on, besides these improvements. Using the binding affinity information, it is obviously possible to apply our general framework to predict quantitative values, using regression models with the same type of kernels. This framework could also be used for a lot of similar problems involving binding, like MHC-type-II-peptide binding where sequences can have variable length and the alignment of epitopes usually performed as pre-processing can be ambiguous. Salomon and Flower (2006) already proposed a kernel for this case. Another interesting application would be drug design, for example protein-kinase-inhibitor binding prediction, or prediction of a virus susceptibility to a panel of drugs for various mutations of the virus.

References

Antes, I., Siu, S. W. I., and Lengauer, T. (2006). DynaPred: a structure and sequence based method for the prediction of MHC class I binding peptide sequences and conformations. *Bioinformatics*, **22**(14), e16–e24.

Aronszajn, N. (1950). Theory of reproducing kernels. *Trans. Am. Math. Soc.*, **68**, 337 – 404.

Bhasin, M. and Raghava, G. P. S. (2004). Prediction of CTL epitopes using QM, SVM and ANN techniques. *Vaccine*, **22**(23-24), 3195–3204.

Bhasin, M., Singh, H., and Raghava, G. P. S. (2003). MHCBN: a comprehensive database of MHC binding and non-binding peptides. *Bioinformatics*, **19**(5), 665–666.

Brusic, V., Petrovsky, N., Zhang, G., and Bajic, V. B. (2002). Prediction of promiscuous peptides that bind HLA class I molecules. *Immunol. Cell Biol.*, **80**(3), 280–285.

Bui, H.-H., Sidney, J., Peters, B., Sathiamurthy, M., Sinichi, A., Purton, K.-A., Mothé, B. R., Chisari, F. V., Watkins, D. I., and Sette, A. (2005). Automated generation and evaluation of specific mhc binding predictive tools: Arb matrix applications. *Immunogenetics*, **57**(5), 304–314.

Bui, H.-H., Schiewe, A. J., von Grafenstein, H., and Haworth, I. S. (2006). Structural prediction of peptides binding to MHC class I molecules. *Proteins*, **63**(1), 43–52.

Buus, S., ller, S. L. L., Worning, P., Kesmir, C., Frimurer, T., Corbet, S., Fomsgaard, A., Hilden, J., Holm, A., and Brunak, S. (2003). Sensitive quantitative predictions of peptide-MHC binding by a 'query by committee' artificial neural network approach. *Tissue Antigens*, **62**(5), 378–384.

Dönnes, P. and Elofsson, A. (2002). Prediction of MHC class I binding peptides, using SVMHC. *BMC Bioinformatics*, **3**(1), 25.

Doytchinova, I. A., Guan, P., and Flower, D. R. (2004). Identifying human MHC supertypes using bioinformatic methods. *J Immunol*, **172**(7), 4314–4323.

Evgeniou, T., Micchelli, C., and Pontil, M. (2005). Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, **6**, 615–637.

Heckerman, D., Kadie, C., and Listgarten, J. (2006). Leveraging information across HLA alleles/supertypes improves HLA-specific epitope prediction.

Hertz, T. and Yanover, C. (2006). PepDist: a new framework for protein-peptide binding prediction based on learning peptide distance functions. *BMC Bioinformatics*, **7 Suppl 1**, S3.

Hertz, T. and Yanover, C. (2007). Identifying hla supertypes by learning distance functions. *Bioinformatics*, **23**(2), e148–e155.

Honeyman, M. C., Brusic, V., Stone, N. L., and Harrison, L. C. (1998). Neural network-based prediction of candidate T-cell epitopes. *Nat. Biotechnol.*, **16**(10), 966–969.

Jojic, N., Reyes-Gomez, M., Heckerman, D., Kadie, C., and Schueler-Furman, O. (2006). Learning MHC I–peptide binding. *Bioinformatics*, **22**(14), e227–e235.

Mamitsuka, H. (1998). Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models. *Proteins*, **33**(4), 460–474.

McMichael, A. and Hanke, T. (2002). The quest for an AIDS vaccine: is the CD8+ T-cell approach feasible? *Nat. Rev. Immunol.*, **2**(4), 283–291.

Milik, M., Sauer, D., Brunmark, A. P., Yuan, L., Vitiello, A., Jackson, M. R., Peterson, P. A., Skolnick, J., and Glass, C. A. (1998). Application of an artificial neural network to predict specific class I MHC binding peptide sequences. *Nat. Biotechnol.*, **16**(8), 753–756.

Nielsen, M., Lundegaard, C., Worning, P., Lauemøller, S. L., Lamberth, K., Buus, S., Brunak, S., and Lund, O. (2003). Reliable prediction of T-cell epitopes using neural networks with novel sequence representations. *Protein Sci.*, **12**(5), 1007–1017.

Parker, K. C., Bednarek, M. A., and Coligan, J. E. (1994). Scheme for ranking potential HLA-A2 binding peptides based on independent binding of individual peptide side-chains. *J. Immunol.*, **152**(1), 163–175.

Peters, B. and Sette, A. (2005). Generating quantitative models describing the sequence specificity of biological processes with the stabilized matrix method. *BMC Bioinformatics*, **6**, 132.

Peters, B., Bui, H.-H., Frankild, S., Nielson, M., Lundegaard, C., Kostem, E., Basch, D., Lamberth, K., Harndahl, M., Fleri, W., Wilson, S. S., Sidney, J., Lund, O., Buus, S., and Sette, A. (2006). A community resource benchmarking predictions of peptide binding to MHC-I molecules. *PLoS Comput Biol*, **2**(6), e65.

Rammensee, H., Bachmann, J., Emmerich, N. P., Bachor, O. A., and Stevanović, S. (1999). Syfpeithi: database for MHC ligands and peptide motifs. *Immunogenetics*, **50**(3-4), 213–219.

Rammensee, H. G., Friede, T., and Stevanoviíc, S. (1995). MHC ligands and peptide motifs: first listing. *Immunogenetics*, **41**(4), 178–228.

Reche, P. A., Glutting, J.-P., and Reinherz, E. L. (2002). Prediction of MHC class I binding peptides using profile motifs. *Hum. Immunol.*, **63**(9), 701–709.

Robinson, J., Malik, A., Parham, P., Bodmer, J. G., and Marsh, S. G. (2000). IMGT/HLA database–a sequence database for the human major histocompatibility complex. *Tissue Antigens*, **55**(3), 280–287.

Rosenfeld, R., Zheng, Q., Vajda, S., and DeLisi, C. (1995). Flexible docking of peptides to class I major-histocompatibility-complex receptors. *Genet. Anal.*, **12**(1), 1–21.

Rötzschke, O., Falk, K., Stevanović, S., Jung, G., and Rammensee, H. C. (1992). Peptide motifs of closely related HLA class I molecules encompass substantial differences. *Eur. J. Immunol.*, **22**(9), 2453–2456.

Salomon, J. and Flower, D. R. (2006). Predicting Class II MHC-Peptide binding: a kernel based approach using similarity scores. *BMC Bioinformatics*, **7**, 501.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA.

Schölkopf, B., Tsuda, K., and Vert, J.-P. (2004). *Kernel Methods in Computational Biology*. MIT Press.

Schueler-Furman, O., Altuvia, Y., Sette, A., and Margalit, H. (2000). Structure-based prediction of binding peptides to MHC class I molecules: application to a broad range of MHC alleles. *Protein Sci.*, **9**(9), 1838–1846.

Sette, A. and Sidney, J. (1998). HLA supertypes and supermotifs: a functional perspective on HLA polymorphism. *Curr Opin Immunol*, **10**(4), 478–482.

Sette, A. and Sidney, J. (1999). Nine major HLA class I supertypes account for the vast preponderance of HLA-A and -B polymorphism. *Immunogenetics*, **50**(3-4), 201–212.

Sette, A., Chesnut, R., and Fikes, J. (2001). HLA expression in cancer: implications for T cell-based immunotherapy. *Immunogenetics*, **53**(4), 255–263.

Sidney, J., del Guercio, M. F., Southwood, S., Engelhard, V. H., Appella, E., Rammensee, H. G., Falk, K., Rötzschke, O., Takiguchi, M., and Kubo, R. T. (1995). Several HLA alleles share overlapping peptide specificities. *J Immunol*, **154**(1), 247–259.

Sidney, J., Grey, H. M., Southwood, S., Celis, E., Wentworth, P. A., del Guercio, M. F., Kubo, R. T., Chesnut, R. W., and Sette, A. (1996). Definition of an HLA-A3-like supermotif demonstrates the overlapping peptide-binding repertoires of common HLA molecules. *Hum Immunol*, **45**(2), 79–93.

Tong, J. C., Zhang, G. L., Tan, T. W., August, J. T., Brusic, V., and Ranganathan, S. (2006). Prediction of HLA-DQ3.2beta ligands: evidence of multiple registers in class II binding peptides. *Bioinformatics*, **22**(10), 1232–1238.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley, New-York.

Wang, R. F. (1999). Human tumor antigens: implications for cancer vaccine development. *J. Mol. Med.*, **77**(9), 640–655.

Yewdell, J. W. and Bennink, J. R. (1999). Immunodominance in major histocompatibility complex class I-restricted T lymphocyte responses. *Annu. Rev. Immunol.*, **17**, 51–88.

Zhang, G. L., Khan, A. M., Srinivasan, K. N., August, J. T., and Brusic, V. (2005). MULTIPRED: a computational system for prediction of promiscuous HLA binding peptides. *Nucleic Acids Res/*, **33**(Web Server issue), W172–W179.

Zhao, Y., Pinilla, C., Valmori, D., Martin, R., and Simon, R. (2003). Application of support vector machines for T-cell epitopes prediction. *Bioinformatics*, **19**(15), 1978–1984.

Zhu, S., Udaka, K., Sidney, J., Sette, A., Aoki-Kinoshita, K. F., and Mamitsuka, H. (2006). Improving MHC binding peptide prediction by incorporating binding data of auxiliary MHC molecules. *Bioinformatics*, **22**(13), 1648–1655.